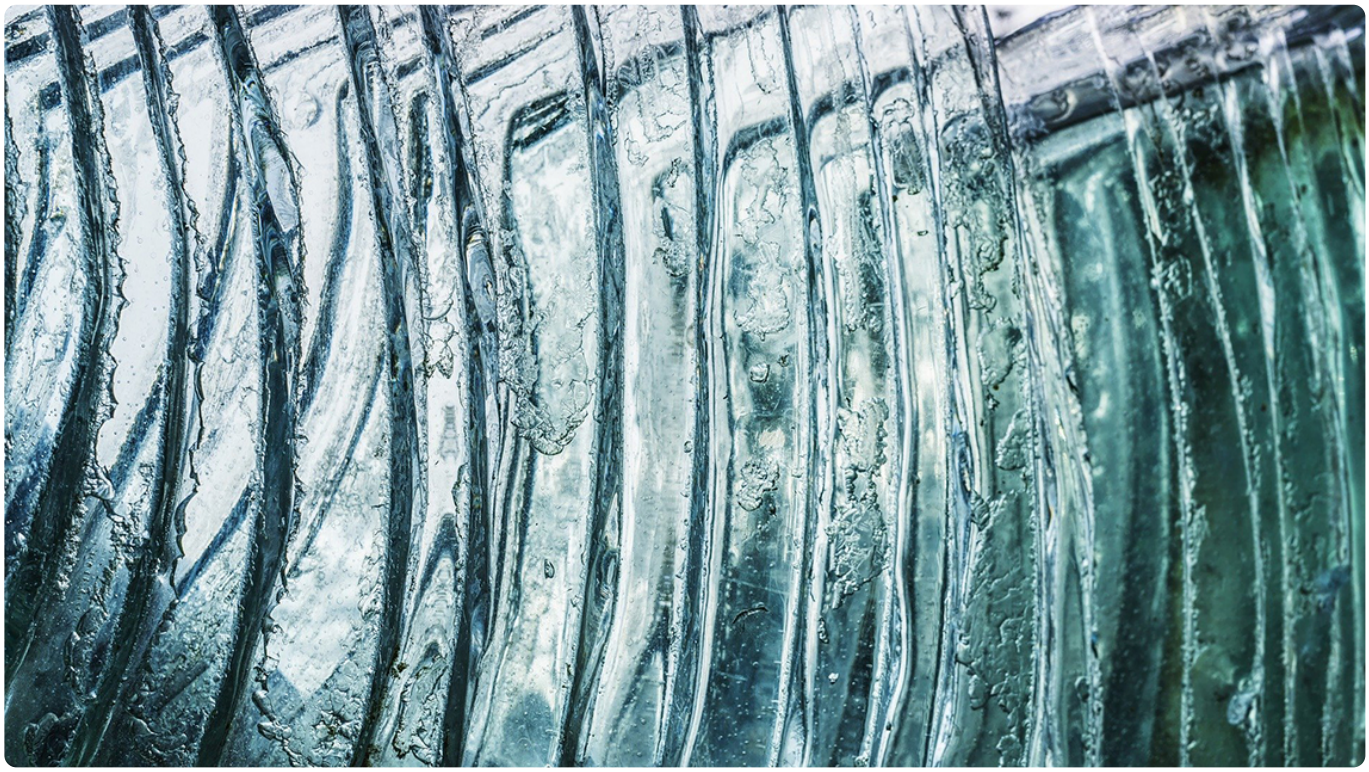
27丨案例：带宽消耗以及Swap（上）

2020-02-21 高楼

性能测试实战30讲 进入课程 >



讲述：高楼
时长 19:19 大小 17.71M

今天我们来看一个真实的案例。事情是这样的，之前有人在微信上问我一个问题，这个问题的现象很典型：典型的 TPS 上不去，响应时间增加，资源用不上。

大概的情况是这样的：有两台 4C8G 的服务器，一台服务器上有 2 个 Tomcat，一台服务器上是 DB。压测的混合场景有 4 个功能模块，其中 3 个访问一个 Tomcat，另外一个访问一个 Tomcat。

Tomcat 的监控页面如下：

Applications | Data Sources | Deployment | Logs | Threads | Cluster | System | Connectors | Certificates | Quick check

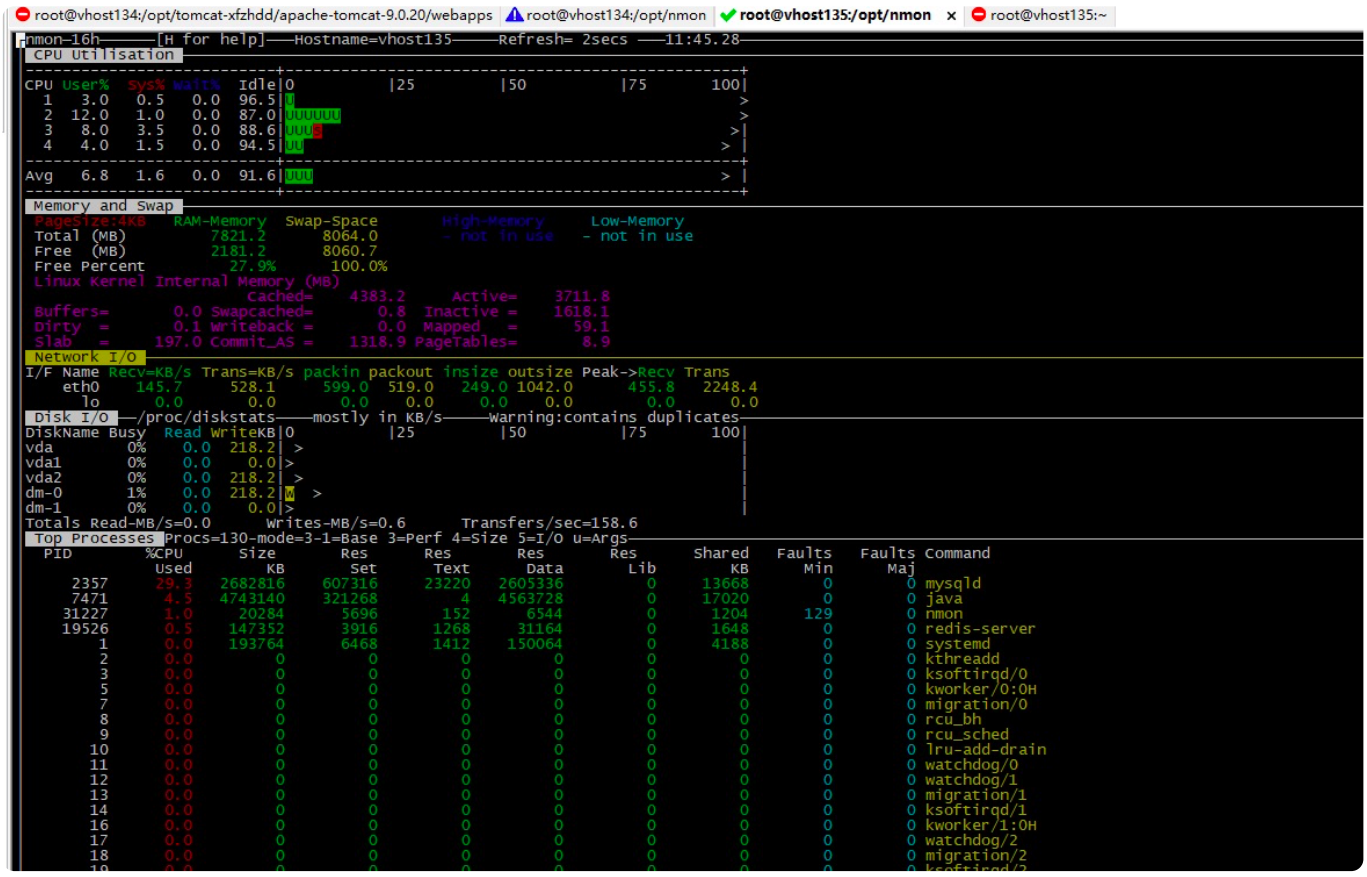
CURRENT MEMORY USAGE

NAME	USAGE SCORE	PLOT	USED	COMMITTED	MAXIMUM	INITIAL	GROUP
Compressed Class Space			12.85 MB	13.88 MB	1.00 GB	0 B	NON_HEAP
Metaspace			115.75 MB	122.06 MB	-1 B	0 B	NON_HEAP
PS Old Gen			75.08 MB	159.00 MB	1.27 GB	83.00 MB	HEAP
PS Survivor Space			272.64 KB	512.00 KB	512.00 KB	5.00 MB	HEAP
PS Eden Space			20.80 MB	32.50 MB	651.00 MB	31.00 MB	HEAP
Code Cache			72.25 MB	73.69 MB	240.00 MB	2.44 MB	NON_HEAP
Total			297.00 MB	401.63 MB	3.14 GB	121.44 MB	TOTAL

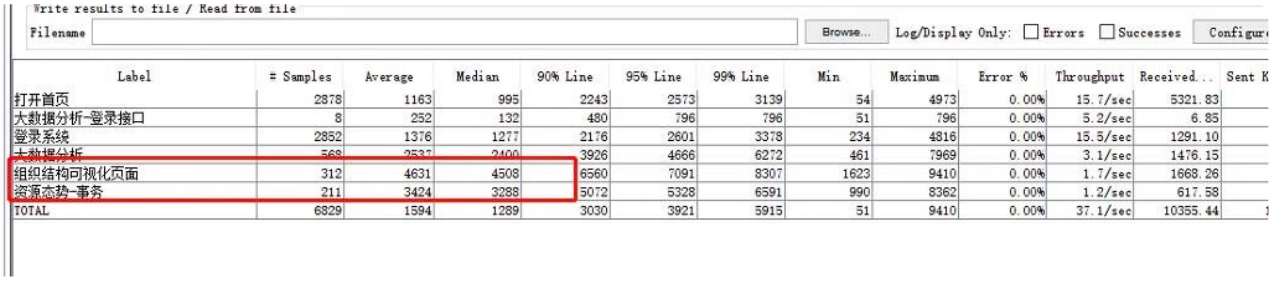
应用服务器□系统资源监控页面如下：

nmon-16h [H for help] Hostname=vhost134 Refresh= 2secs 11:45.15
CPU Utilisation
CPU User% Sys% Wait% Idle
1 20.1 5.0 0.0 74.9
2 23.7 2.5 0.0 73.7
3 25.0 11.5 0.0 63.5
4 24.7 5.1 0.0 70.2
Avg 23.3 6.1 0.0 70.6
Memory and Swap
PageSize:4KB RAM-Memory Swap-Space High-Memory Low-Memory
Total (MB) 7821.3 8064.0 - not in use - not in use
Free (MB) 3514.7 7860.7
Free Percent 44.9% 97.5%
Linux Kernel Internal Memory (MB)
Cached= 2060.8 Active= 3386.7
Buffers= 0.0 Swapcached= 3.6 Inactive = 672.8
Dirty = 4.8 Writeback = 0.0 Mapped = 48.0
Slab = 123.3 Commit_AS = 3689.3 PageTables= 15.9
Network I/O
I/F Name Recv=KB/s Trans=KB/s packin packout insize outsize Peak->Recv Trans
eth0 928.4 11656.2 5124.0 5051.6 185.5 2362.8 1274.7 11869.9
lo 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
Disk I/O /proc/diskstats mostly in KB/s Warning:contains duplicates
DiskName Busy Read WriteKB
vda 0% 0.0 0.0
vda1 0% 0.0 0.0
vda2 0% 0.0 0.0
dm-0 0% 0.0 0.0
dm-1 0% 0.0 0.0
Totals Read-MB/s=0.0 Writes-MB/s=0.0 Transfers/sec=0.0
Top Processes Procs=154-mode=3-1=Base 3=Perf 4=Size 5=I/O u=Args
PID %CPU Used Size KB Res Set Res Text Res Data Res Lib Shared KB Faults Min Faults Maj Command
20149 109.2 5953048 1399668 4 5759516 0 15408 58 0 java
6703 5.5 5828608 609852 4 5635076 0 15472 0 0 java
20864 1.0 20300 5720 152 6560 0 1204 152 0 nmon
3092 0.5 90392 2084 36 728 0 2028 0 0 rngd
23950 0.5 0 0 0 0 0 0 0 0 kworker/0:0
1 0.0 193788 4652 1412 150088 0 2976 0 0 systemd
2 0.0 0 0 0 0 0 0 0 0 kthreadd
3 0.0 0 0 0 0 0 0 0 0 ksoftirqd/0
5 0.0 0 0 0 0 0 0 0 0 kworker/0:0H
7 0.0 0 0 0 0 0 0 0 0 migration/0
8 0.0 0 0 0 0 0 0 0 0 rcu_bh
9 0.0 0 0 0 0 0 0 0 0 rcu_sched
10 0.0 0 0 0 0 0 0 0 0 lru-add-drain
11 0.0 0 0 0 0 0 0 0 0 watchdog/0
12 0.0 0 0 0 0 0 0 0 0 watchdog/1
13 0.0 0 0 0 0 0 0 0 0 migration/1
14 0.0 0 0 0 0 0 0 0 0 ksoftirqd/1

数据库服务器系统资源监控如下：



JMeter 结果如下：



综上现象就是，单业务场景执行起来并不慢，但是一混合起来就很慢，应用服务器和数据库服务器的系统资源使用率并不高。请问慢在哪？

这是非常典型的询问性能问题的方式，虽然多给了系统资源信息，但是这些信息也不足以说明瓶颈在哪。

为什么呢？在现在多如牛毛的监控工具中，除非我们在系统中提前做好分析算法或警告，否则不会有监控工具主动告诉你， 监控出的某个数据有问题，而这个只能靠做性能分析的人来判断。

我在很多场合听一些“专家”说：性能分析判断要遵守木桶原理。但是在做具体技术分析的时候，又不给人说明白木桶的短板在哪里。这就好像，一个□赛车手说要是有一个各方面都好的车，我肯定能得第一名，但是，我没有车。

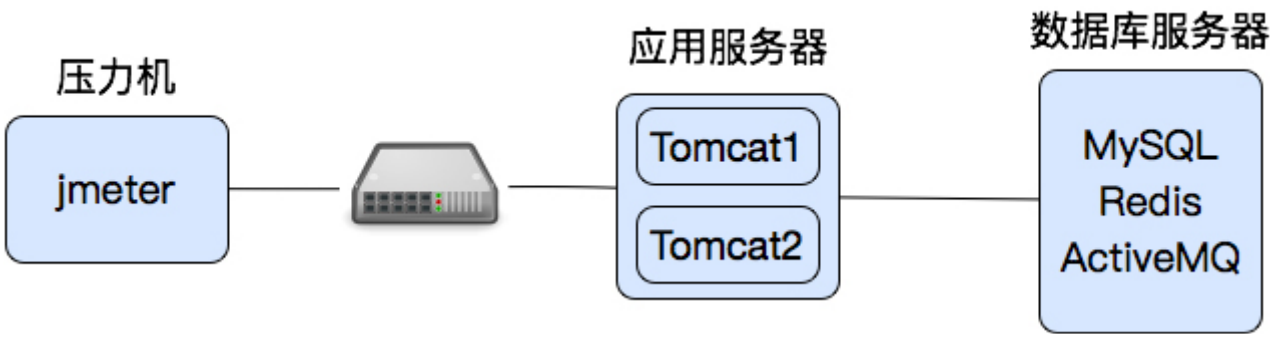
话说出来轻而易举，但是请问木桶的短板怎么判断呢？因为 CPU 高，所以 CPU 就是短板吗？所以就要加 CPU 吗？这肯定是不对的。

因为这个例子并不大，所以可以细细地写下去。今天文章的目的就是要告诉你，性能问题分析到底应该是个什么分析思路。

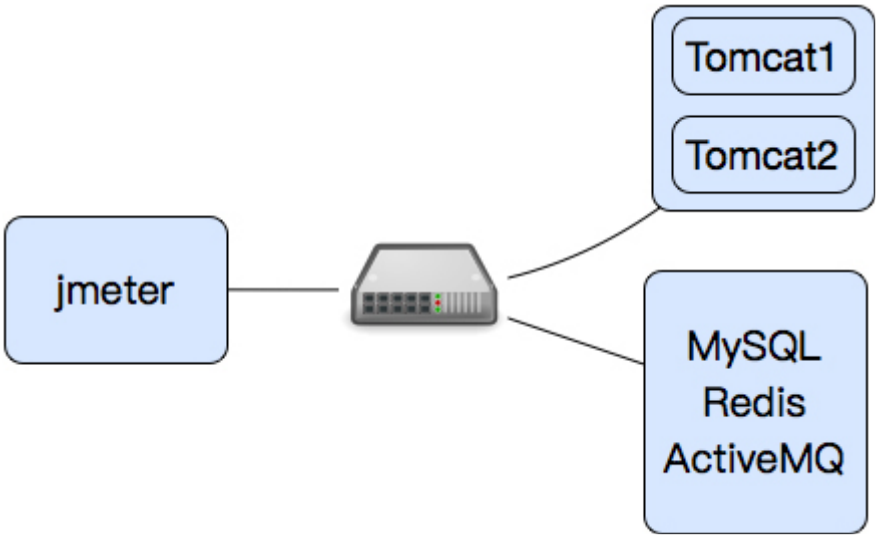
分析的第一阶段

画架构图

做性能测试时，我们需要先画一个架构图，虽然简单，但是让自己脑子里时时记得架构图，是非常有必要的。因为架构级的分析，就是要胸怀架构，在看到一个问题点时，可以从架构图中立即反应出来问题的相关性。



上面这张图是自己脑子里的逻辑图，数据在网络中的流转并不是这样，而是像下图这样。



数据流是从压力机到应用服务器，应用服务器再到网络设备，再到数据库服务器；数据库把数据返回给应用服务器，应用服务器再通过网络设备给压力机。

如果把里面的 Redis、ActiveMQ 和 MySQL 的逻辑再细说明白，那么这个小小的应用都可以描述好一会。所以这里，我先大概描述一下，如果后面的分析中涉及到了相应的逻辑，再一点点加进来。

应用服务器只有一台，上面有两个 Tomcat 实例；数据库服务器有三个应用。混合场景中有四个业务，其中三个访问 Tomcat1，第四个访问 Tomcat2。

场景□描述

有了场景大概的画像之后，我们再来看场景。根据测试工程师描述：

1. 响应时间慢的，都是可视化页面，有不少图片、JS、CSS 等静态资源。公网上是用 CDN 的，现在只测试内网的部分。静态资源已经做了压缩。
2. 单业务测试的容量是可以满足要求的，但混合场景响应时间就长。系统资源用得并不多。
3. 压力场景是 300 线程，Ramp-up period 是 1 秒。
4. Duration 是 72000。
5. 各参数化都已经做了，参数化数据也合理。
6. 测试环境都是内网。
7. 服务器是 CentOS，压力机是 Win10。

既然这样，我们还是要看看系统的各个资源，再来判断方向。我在很多场合都强调证据链。对架构比较简单的应用来说，我们都不用再做时间的拆分了，直接到各主机上看看资源就好了。

瓶颈□分析定位

根据我们之前画的架构图，我们从应用服务器、数据库服务器和压力数据分别定位一下。

应用服务器

从前面的应用服务器资源来看，CPU 使用率现在还不高，也基本都在 us CPU（就是 user 消耗的 CPU）上，比例也比较合理。

物理内存 8G，还有 3.5G。即使不多，对 Java 应用来说，也要先看 JVM，只要不是 page fault 过多，我们可以先不用管物理内存。

网络资源接收 900KB/s 左右，发送 11M 左右。这样的带宽应该说是比较敏感的，因为对 100Mbps 和 1000Mbps 来说，我们要心里有一个数，一个是 12.5MB（对应 100Mbps），一个是 125MB（对应 1000Mbps），当和这样的带宽值接近的时候，就要考虑下是不是带宽导致的压力上不去。不过这里我们也不用先下定论。只是留个疑问在这里。

磁盘资源，基本上没有读写，很正常。

从 Process 列表中，也没看到什么异常数据。faults 也都是 min，major fault 并没有。但在这个案例中，还部署了另一个监控工具，上面显示如下：

为什么这个 Swapping 要标黄呢？那肯定是过大了嘛。是的，你可以觉得 swap 过多。这个扣，我们也记在心里。在这里标红加粗敲黑板！

数据库服务器

照样分析，CPU、内存、网络、磁盘、Process 列表，并没看到哪里有异常的数据，连网络都只有 500 多 k 的发送。

这样的数据库资源状态□告诉我们，它不在我们的问题分析主线上。接下来是压力数据。

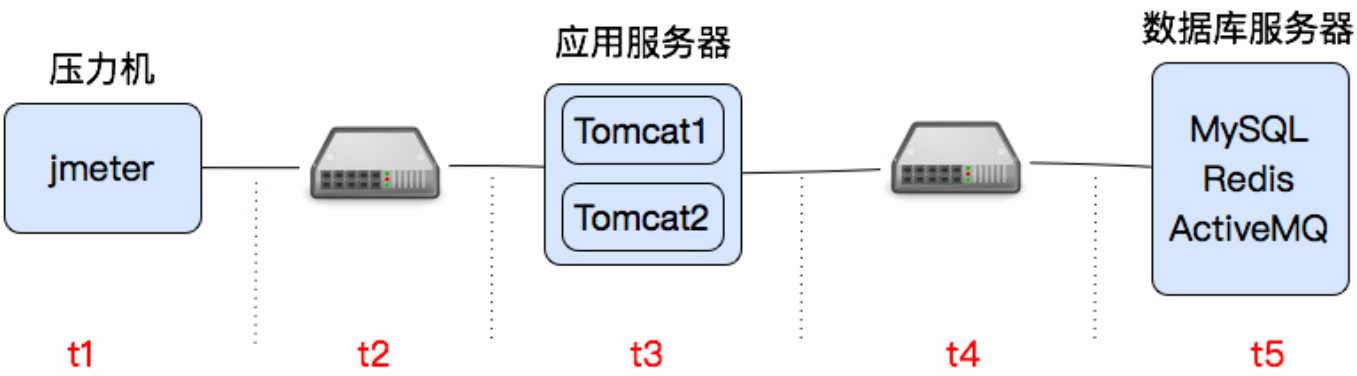
压力数据

这是 JMeter 中的聚合报告：

# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received...	Sent K
2878	1163	995	2243	2573	3139	54	4973	0.00%	15.7/sec	5321.83	
8	252	132	480	796	796	51	796	0.00%	5.2/sec	6.85	
2852	1376	1277	2176	2601	3378	234	4816	0.00%	15.5/sec	1291.10	
568	2537	2400	3926	4666	6272	461	7969	0.00%	3.1/sec	1476.15	
312	4631	4508	6560	7091	8307	1623	9410	0.00%	1.7/sec	1668.26	
211	3424	3288	5072	5328	6591	990	8362	0.00%	1.2/sec	617.58	
6829	1594	1289	3030	3921	5915	51	9410	0.00%	37.1/sec	10355.44	

从上面这张图也能看出，响应时间确实挺长的，并且，□300 线程只有 37 的 TPS，带宽总量 10M 左右。这个带宽倒是和应用服务器上的带宽使用量相当。关于这个带宽的判断请你一定注意，对于性能分析来说，带宽能不能对得上非常重要。□比如，客户端接收了多少流量，服务端就应该是发出了多少流量。如果服务端发了很多包，但是客户端没有接收，那就是堵在队列上了。

既然其它的资源暂时没出现什么瓶颈。其实在这个时间里，如果是复杂的应用的话，我们最应该干的一件事情就是拆分时间。如下图所示：



这里我把时间拆为 t1-t5，具体分析哪一段为什么消耗了时间。我们可以在 Tomcat 中加上 %D 和 %F 两个参数来记录 request 和 response 的时间。

□在没有做这个动作之前，我们先把前面的扣解决一下。首先，带宽是不是受了 100Mbps 的限制?

一般来说，判断网络的时候，我们会有几个判断点。

首先是带宽的流量大小，也就是前面我们看到的 11M 左右的值。一般来说，100Mbps 是指的 bit per second，但是在应用层基本上都是 byte，所以对 100Mbps 来说，是 12.5MB。

其次是，全连接和半连接队列是否已经溢出?

```
21 times the listen queue of a socket overflowed
27 SYNs to LISTEN sockets dropped
```

我们通过 SYNs to LISTEN sockets dropped 来判断半连接队列是否溢出，通过 times the listen queue of a socket overflowed 来判断全连接队列是否溢出。

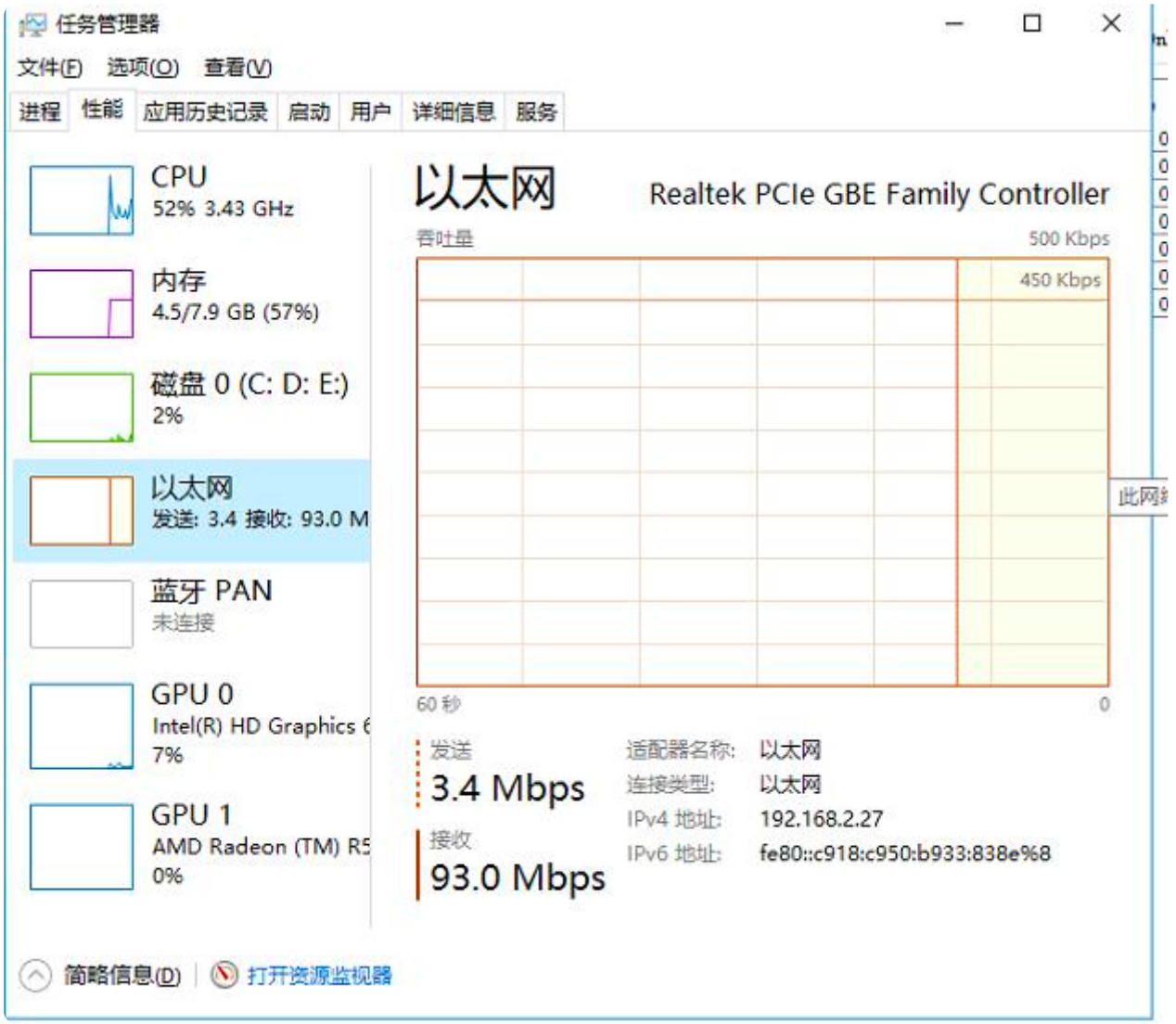
通过实时的查看，这两个值的增加并不多。所以这里不会是问题点。

最后是发送和接收队列是否堆积?

```
tcp6       0    214 192.168.1.134:38356     192.168.1.135:3306      ESTABLISHED 1184/java       keepalive (7092.42/0/0)
tcp6       0    214 192.168.1.134:38242     192.168.1.135:3306      ESTABLISHED 1184/java       keepalive (6109.38/0/0)
tcp6       0      0 192.168.1.134:8082      192.168.2.27:51053      ESTABLISHED 1184/java       off (0.00/0/0)
tcp6       0    192 192.168.1.134:38322     192.168.1.135:3306      ESTABLISHED 1184/java       keepalive (7076.03/0/0)
tcp6       0    204 192.168.1.134:38270     192.168.1.135:3306      ESTABLISHED 1184/java       keepalive (7043.26/0/0)
tcp6       0    576 192.168.1.134:38326     192.168.1.135:3306      ESTABLISHED 1184/java       keepalive (7076.03/0/0)
tcp6       0    214 192.168.1.134:38372     192.168.1.135:3306      ESTABLISHED 1184/java       keepalive (7092.41/0/0)
tcp6       0  74123 192.168.1.134:8082      192.168.2.27:51307      ESTABLISHED 1184/java       off (0.00/0/0)
tcp6       0      0 192.168.1.134:8082      192.168.2.27:51009      ESTABLISHED 1184/java       off (0.00/0/0)
tcp6       0  51650 192.168.1.134:8082      192.168.2.27:51200      ESTABLISHED 1184/java       on (0.47/0/0)
tcp6       0   2756 192.168.1.134:8082      192.168.2.27:51301      ESTABLISHED 1184/java       off (0.00/0/0)
tcp6       0    877 192.168.1.134:38398     192.168.1.135:3306      ESTABLISHED 1184/java       keepalive (7125.18/0/0)
tcp6       0    214 192.168.1.134:38396     192.168.1.135:3306      ESTABLISHED 1184/java       keepalive (7125.18/0/0)
tcp6       0  18967 192.168.1.134:8082      192.168.2.27:51316      ESTABLISHED 1184/java       off (0.00/0/0)
tcp6       0      0 192.168.1.134:8082      192.168.2.27:51179      ESTABLISHED 1184/java       off (0.00/0/0)
tcp6       0    210 192.168.1.134:38294     192.168.1.135:3306      ESTABLISHED 1184/java       keepalive (7059.65/0/0)
tcp6       0      0 192.168.1.134:8082      192.168.2.27:51138      ESTABLISHED 1184/java       off (0.00/0/0)
tcp6       0  24026 192.168.1.134:8082      192.168.2.27:51153      ESTABLISHED 1184/java       off (0.00/0/0)
tcp6       0      0 192.168.1.134:8082      192.168.2.27:51025      ESTABLISHED 1184/java       off (0.00/0/0)
tcp6       0   2143 192.168.1.134:8082      192.168.2.27:51161      ESTABLISHED 1184/java       off (0.00/0/0)
tcp6       0   2756 192.168.1.134:8082      192.168.2.27:51304      ESTABLISHED 1184/java       off (0.00/0/0)
tcp6       0   2331 192.168.1.134:8082      192.168.2.27:51229      ESTABLISHED 1184/java       off (0.00/0/0)
tcp6       0      0 192.168.1.134:8081      192.168.2.27:51113      ESTABLISHED 6703/java       off (0.00/0/0)
tcp6       0  48730 192.168.1.134:8082      192.168.2.27:51246      ESTABLISHED 1184/java       on (0.35/0/0)
tcp6       0   1282 192.168.1.134:8082      192.168.2.27:51132      ESTABLISHED 1184/java       off (0.00/0/0)
tcp6       0      0 192.168.1.134:8082      192.168.2.27:51137      ESTABLISHED 1184/java       off (0.00/0/0)
tcp6       0    877 192.168.1.134:38276     192.168.1.135:3306      ESTABLISHED 1184/java       keepalive (7059.65/0/0)
tcp6       0  38205 192.168.1.134:8082      192.168.2.27:51006      ESTABLISHED 1184/java       off (0.00/0/0)
tcp6       0      0 192.168.1.134:8082      192.168.2.27:51169      ESTABLISHED 1184/java       off (0.00/0/0)
tcp6       0      0 192.168.1.134:8081      192.168.8.167:60128     ESTABLISHED 6703/java       off (0.00/0/0)
tcp6       0      0 192.168.1.134:8081      192.168.2.27:51163      ESTABLISHED 6703/java       off (0.00/0/0)
tcp6       0  45117 192.168.1.134:8082      192.168.2.27:51136      ESTABLISHED 1184/java       off (0.00/0/0)
tcp6       0      0 192.168.1.134:8082      192.168.2.27:51218      ESTABLISHED 1184/java       off (0.00/0/0)
tcp6       0    210 192.168.1.134:38256     192.168.1.135:3306      ESTABLISHED 1184/java       keepalive (7026.88/0/0)
tcp6       0      0 192.168.1.134:8082      192.168.94.114:9630     ESTABLISHED 1184/java       off (0.00/0/0)
tcp6       0   1992 192.168.1.134:8082      192.168.2.27:51249      ESTABLISHED 1184/java       off (0.00/0/0)
tcp6       0  49282 192.168.1.134:8082      192.168.2.27:51065      ESTABLISHED 1184/java       off (0.00/0/0)
tcp6       0      0 192.168.1.134:8082      192.168.2.27:51382      ESTABLISHED -               off (0.00/0/0)
tcp6       0   4367 192.168.1.134:8082      192.168.2.27:51274      ESTABLISHED 1184/java       off (0.00/0/0)
tcp6       0   3287 192.168.1.134:8082      192.168.2.27:50995      ESTABLISHED 1184/java       off (0.00/0/0)
tcp6       0  11057 192.168.1.134:8082      192.168.2.27:51116      ESTABLISHED 1184/java       off (0.00/0/0)
tcp6       0    193 192.168.1.134:38400     192.168.1.135:3306      ESTABLISHED 1184/java       keepalive (7125.18/0/0)
```

通过应用服务器上的 send-Q（前面数第三列），可以看到服务器和压力机之间的的队列还是很长的，基本上每次查看都存在，这说明队列一直都有堆积。

我们再到压力机上看看带宽用了多少：



看这里也是用到了 93Mbps，那么到这里我们就可以确定是网络问题导致的 TPS 上不去，响应时间增加，系统资源也用不上了。

和系统管理员确认宿主机的带宽后，被告知宿主机确实是 100Mbps。

似乎这个分析到这里就可以结束了，直接把带宽加上再接着测试呗。但是，从项目实施的角度上说，这个问题，并不是阻塞性的。

为了把更多的性能问题提前找出来，现在我们先不下载静态□资源，只发接口请求找下其他□□性能问题。这个带宽的问题，记一个 bug 就行了。

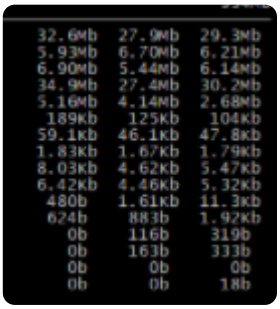
优化结果

我们将静态资源全都过滤掉之后，再次执行场景，结果是下面这样的。

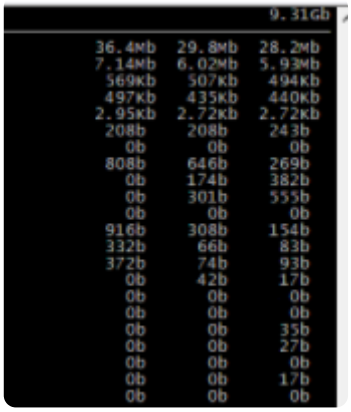
JMeter 压力数据：

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received...	Sent KB/sec
视化页面	3037	4206	3689	6074	8055	11041	1649	15421	0.00%	9.4/sec	1504.98	4.69
事务	1998	3199	3369	4624	7545	12392	151	18290	0.00%	6.2/sec	499.42	6.90
	23760	2706	2324	3896	5864	8425	97	14467	0.00%	73.1/sec	604.66	73.64
登录接口	40	577	381	1448	1479	1512	21	1512	0.00%	16.8/sec	22.35	2.80
	43197	280	207	309	364	2238	44	22165	0.00%	141.0/sec	848.54	429.02
	72032	1327	279	3242	4174	8001	21	22165	0.00%	221.5/sec	3391.42	489.42

应用服务器带宽：



数据库服务器带宽：



应用服务器网络队列：

```
tcp6       0      0 192.168.1.134:8082      192.168.2.27:43807      ESTABLISHED 1184/java           off (0.00/0/0)
tcp6       0      0 192.168.1.134:8082      192.168.2.27:43984      ESTABLISHED 1184/java           off (0.00/0/0)
tcp6       0      0 192.168.1.134:8082      192.168.2.27:44042      ESTABLISHED 1184/java           off (0.00/0/0)
tcp6       0      0 192.168.1.134:8081      192.168.2.27:44063      ESTABLISHED 6703/java           off (0.00/0/0)
tcp6       0      0 192.168.1.134:8082      192.168.8.167:60481     ESTABLISHED 1184/java           off (0.00/0/0)
tcp6       0      0 192.168.1.134:8082      192.168.2.27:43667      ESTABLISHED 1184/java           off (0.00/0/0)
tcp6       0      0 192.168.1.134:38622     192.168.1.135:3306      ESTABLISHED 6703/java           keepalive (6734.39/0/0)
tcp6       0      0 192.168.1.134:8082      192.168.2.27:43410      ESTABLISHED 1184/java           off (0.00/0/0)
tcp6       0      0 192.168.1.134:38714     192.168.1.135:3306      ESTABLISHED 1184/java           keepalive (6734.39/0/0)
tcp6       0      0 192.168.1.134:8082      192.168.2.27:43272      ESTABLISHED 1184/java           off (0.00/0/0)
tcp6       0      0 192.168.1.134:38552     192.168.1.135:3306      ESTABLISHED 1184/java           keepalive (6718.01/0/0)
tcp6       0      0 192.168.1.134:38570     192.168.1.135:3306      ESTABLISHED 6703/java           keepalive (6718.01/0/0)
tcp6       0      0 192.168.1.134:38530     192.168.1.135:3306      ESTABLISHED 1184/java           keepalive (6718.01/0/0)
tcp6       0      0 192.168.1.134:38554     192.168.1.135:3306      ESTABLISHED 1184/java           keepalive (6718.01/0/0)
tcp6       0      0 192.168.1.134:8082      192.168.94.114:12830    ESTABLISHED 1184/java           off (0.00/0/0)
tcp6       0      0 192.168.1.134:8082      192.168.2.27:43953      ESTABLISHED 1184/java           off (0.00/0/0)
tcp6       0      0 192.168.1.134:8082      192.168.2.27:43555      ESTABLISHED 1184/java           off (0.00/0/0)
tcp6       0      0 192.168.1.134:8082      192.168.2.27:43752      ESTABLISHED 1184/java           off (0.00/0/0)
tcp6       0      0 192.168.1.134:8082      192.168.2.27:43503      ESTABLISHED 1184/java           off (0.00/0/0)
tcp6       0      0 192.168.1.134:8082      192.168.2.27:43763      ESTABLISHED 1184/java           off (0.00/0/0)
tcp6       0      0 192.168.1.134:8082      192.168.2.27:43493      ESTABLISHED 1184/java           off (0.00/0/0)
tcp6       0      0 192.168.1.134:8081      192.168.2.27:44083      ESTABLISHED 6703/java           off (0.00/0/0)
tcp6       0      0 192.168.1.134:38758     192.168.1.135:3306      ESTABLISHED 1184/java           keepalive (6750.78/0/0)
tcp6       0      0 192.168.1.134:8082      192.168.2.27:43199      ESTABLISHED 1184/java           off (0.00/0/0)
tcp6       0      0 192.168.1.134:8082      192.168.2.27:43809      ESTABLISHED 1184/java           off (0.00/0/0)
tcp6       0      0 192.168.1.134:8082      192.168.2.27:43456      ESTABLISHED 1184/java           off (0.00/0/0)
tcp6       0      0 192.168.1.134:8081      192.168.94.139:53025    ESTABLISHED 6703/java           off (0.00/0/0)
tcp6       0      0 192.168.1.134:38534     192.168.1.135:3306      ESTABLISHED 1184/java           keepalive (6718.01/0/0)
tcp6       0      0 192.168.1.134:8082      192.168.2.27:44084      ESTABLISHED 1184/java           off (0.00/0/0)
tcp6       0      0 192.168.1.134:38544     192.168.1.135:3306      ESTABLISHED 1184/java           keepalive (6718.01/0/0)
tcp6       0    500 192.168.1.134:8081      192.168.2.27:44104      ESTABLISHED 6703/java           off (0.00/0/0)
tcp6       0      0 192.168.1.134:8082      192.168.2.27:43560      ESTABLISHED 1184/java           off (0.00/0/0)
tcp6       0      0 192.168.1.134:38586     192.168.1.135:3306      ESTABLISHED 6703/java           keepalive (6718.01/0/0)
tcp6       0      0 192.168.1.134:8082      192.168.2.27:43472      ESTABLISHED 1184/java           off (0.00/0/0)
tcp6       0      0 192.168.1.134:8082      192.168.2.27:43509      ESTABLISHED 1184/java           off (0.00/0/0)
tcp6       0      0 192.168.1.134:38604     192.168.1.135:3306      ESTABLISHED 6703/java           keepalive (6718.01/0/0)
tcp6       0      0 192.168.1.134:38588     192.168.1.135:3306      ESTABLISHED 6703/java           keepalive (6718.01/0/0)
tcp6       0      0 192.168.1.134:38726     192.168.1.135:3306      ESTABLISHED 1184/java           keepalive (6734.39/0/0)
tcp6       0      0 192.168.1.134:38750     192.168.1.135:3306      ESTABLISHED 1184/java           keepalive (6750.78/0/0)
tcp6       0      0 192.168.1.134:38684     192.168.1.135:3306      ESTABLISHED 1184/java           keepalive (6734.39/0/0)
tcp6       0      0 192.168.1.134:8082      192.168.2.27:44116      ESTABLISHED -                   off (0.00/0/0)
tcp6       0      0 192.168.1.134:38532     192.168.1.135:3306      ESTABLISHED 1184/java           keepalive (6718.01/0/0)
```

应用服务器资源监控：

通过上面的结果可以看出：

1. TPS 可以达到 221.5 了，并且 Received 和 Sent 的字节加一起不到 4MB。

2. 应用服务器和数据库服务器的带宽都用到了近 40Mbps，和 JMeter 结果也相当。

3. 应用服务器上的网络队列也没有堆积。

4. 应用服务器的 CPU 也已经能消耗到 66% 了，

当正在想通过过滤掉静态资源绕过带宽不足的现状来测试其他性能问题的时候，这时，Swap 双向都标黄了。这时，性能测试工程师更纠结了，它为什么双向都黄了？CPU 使用率才 66% 嘛。

其实，这两句话之间并没有什么关系，CPU 使用率不管是多少，Swap 该黄还是会黄。

这是为什么呢？这里卖个关子，在下一篇文章中，我们接着分析。

总结

带宽问题是性能分析中常见的问题之一，其难点就在于，带宽不像 CPU 使用率那么清晰可理解，它和 TCP/IP 协议的很多细节有关，像三次握手，四次挥手，队列长度，网络抖动、丢包、延时等等，都会影响性能，关键是这些判断点还不在同一个界面中，所以需要做分析的人有非常明确的分析思路才可以做得到。而且现在的监控分析工具中，对网络的判断也是非常薄弱的。

而 Swap 问题不能算是常见，只要出现，基本上就会很多人晕乎。解决的关键就是要明白 Swap 的原理，查到关联参数，然后就可以很快地定位了。

思考题

结合今天的内容，你能说一下网络的瓶颈如何判断吗？有哪几个队列？

欢迎你在评论区写下你的思考，也欢迎把这篇文章分享给你的朋友或者同事，一起交流一下。

课程学习计划

关注极客时间服务号
每日学习签到

月领 25+ 极客币

【点击】保存图片，打开【微信】扫码>>>

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇　26 | 案例：手把手带你理解TPS趋势分析

下一篇　28 | 案例：带宽消耗以及Swap（下）

精选留言 (2)

写留言

Melody
2020-02-25

老师，有一个问题想请教您，服务器使用的是容器，且做了负载均衡，在加压的过程中会报连接超时的错误，服务器cpu，内存，带宽都正常，压力机也未出现瓶颈，该如何分析？

作者回复: 网络呢?

大拇哥
2020-02-21

1）查看服务器发包数量，客户端接口数据包的量是不基本一致，如果出现明显的差别，可以基本确定客户端网络问题
2）查看jmeter聚会报告上面客户端接口数据量是否已经接近宽带的限制
队列：

服务端消息发送队列...

展开

作者回复: 1，当网络有问题的时候，主要看队列，不用再对比流量了，因为已经没有意义了。
2，对。